Amendments to the Claims:

This listing of claims will replace all prior versions and listings of claims in this application:

Listing of Claims:

1. (Currently amended) A processor-implemented method of federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

adding a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

allowing local applications to access both the data exposed in the local file system and data in other parts of the distributed file system; ~~and~~

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

assigning an object ID number to each object found in the local file system;

generating the object ID number as a tuple <I,G> of an inode number of the object found in the local file system, and a generation number;

wherein the generation number is increased anytime a same inode number is found for a different object; and

maintaining a deleted flag for objects that have been deleted, so the occurrence of the same inode number for different objects can be recognized.

2. (Original) The method of claim 1, wherein the federation layer comprises a virtual server that serves file data and metadata from the local file system to the distributed file system.

3. (Original) The method of claim 2, wherein the virtual server comprises:

a virtual metadata server that serves file metadata;

a virtual storage server that serves file content; and

wherein the virtual metadata and the virtual storage server serve data from the local file system to the distributed file system.

4. (Original) The method of claim 3, wherein the virtual storage server is a virtual object storage server.

5. (Canceled)

6. (Original) The method of claim 2, wherein the federation layer contains a distributed file system client for transferring and translating communications from a local application on the local computer system to the virtual server.

7. (Currently amended) The method of claim 2, further comprises installing the ~~virtual~~ file server in the local computer system.

8. (Original) The method of claim 3, further comprises installing the virtual metadata server in the local computer system.

9. (Original) The method of claim 4, further comprises installing the virtual object storage server in the local computer system.

10. (Original) The method of claim 6, further comprises installing the distributed file system client in the local computer system.

11. (Currently amended) The method of claim 2, further comprises configuring the ~~virtual~~ file server to communicate with the distributed file system.

12. (Original) The method of claim 11, further comprises:

temporarily disconnecting the local applications on the local computer system; and

reconfiguring the local applications to communicate with the distributed file system client.

13. (Canceled)

14. (Currently amended) The method of claim [[13]] 1, further comprises generating the object ID number by counting the objects that were found in the local file system.

15. (Canceled)

16. (Currently amended) A processor-implemented method of federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

adding a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

allowing local applications to access both the data exposed in the local file system and data in other parts of the distributed file system;

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

assigning an object ID number to each object found in the local file system;

~~The method of claim 13, further comprises:~~

detecting multiple hard links to a same file by comparing an inode number of files;

if one file has more than one hard link, using a same object ID number for the multiple names for that object; and

deleting only the object when a last hard link has been unlinked.

17. (Original) The method of claim 6, further comprises using a shared memory between the distributed file system client and the virtual server to enhance communication between the distributed file system client and the virtual server.

18. (Currently amended) The ~~system~~ method of claim 6, further comprises moving the virtual server into a kernel space for efficiency purpose.

19. (Currently amended) A computer program product having instruction codes stored on a computer-usable medium for federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

a ~~first~~ set of instruction codes for adding a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

a ~~second~~ set of instruction codes for allowing local applications to access both the data exposed in the local file system and data in other parts of the distributed file system; and

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

a set of instruction codes for assigning an object ID number to each object found in the local file system;

a set of instruction codes for generating the object ID number as a tuple <I,G> of an inode number of the object found in the local file system, and a generation number;

a set of instruction codes for increasing the generation number anytime a same inode number is found for a different object; and

a set of instruction codes for maintaining a deleted flag for objects

that have been deleted, so the occurrence of the same inode number for different objects can be recognized.

20. (Original) The computer program product of claim 19, wherein the federation layer comprises a virtual server that serves file data and metadata from the local file system to the distributed file system.

21. (Original) The computer program product of claim 20, wherein the virtual server comprises:
a virtual metadata server that serves file metadata;
a virtual storage server that serves file content; and
wherein the virtual metadata and the virtual storage server serve data from the local file system to the distributed file system.

22. (Original) The computer program product of claim 21, wherein the virtual storage server is a virtual object storage server.

23. (Canceled)

24. (Original) The computer program product of claim 20, wherein the federation layer contains a distributed file system client for transferring and translating communications from a local application on the local computer system to the virtual server.

25. (Currently amended) The computer program product of claim 20, further comprises a third set of instruction codes for installing the ~~virtual~~ file

server in the local computer system.

26. (Currently amended) The computer program product of claim 21, further comprises a ~~fourth~~ set of instruction codes for installing the virtual metadata server and the virtual file in the local computer system.

27. (Currently amended) The computer program product of claim 22, further comprises a ~~fifth~~ set of instruction codes for installing the virtual object storage server in the local computer system.

28. (Currently amended) The computer program product of claim 23, further comprises a ~~sixth~~ set of instruction codes for installing the distributed file system client in the local computer system.

29. (Currently amended) The computer program product of claim 20, further comprises a ~~seventh~~ set of instruction codes for configuring the ~~virtual~~ file server to communicate with the distributed file system.

30. (Currently amended) The computer program product of claim 29, further comprises:

~~an eight~~ a set of instruction codes for temporarily disconnecting the local applications on the local computer system; and

a ~~ninth~~ set of instruction codes for reconfiguring the local applications to communicate with the distributed file system client.

31. (Currently amended) A processor-implemented service for federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

an addition of a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

an allowance of local applications to access both the data exposed in the local file system and data in other parts of the distributed file system; ~~and~~

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

an assignment of an object ID number to each object found in the local file system;

a generation of the object ID number as a tuple <I,G> of an inode number of the object found in the local file system, and a generation number;

wherein the generation number is increased anytime a same inode number is found for a different object; and

a maintenance of a deleted flag for objects that have been deleted, so the occurrence of the same inode number for different objects can be recognized.

32. (Original) The service of claim 31, wherein the federation layer comprises a virtual server that serves file data and metadata from the local file system

to the distributed file system.

33. (Original) The service of claim 32, wherein the virtual server comprises:

a virtual metadata server that serves file metadata;

a virtual storage server that serves file content; and

wherein the virtual metadata and the virtual storage server serve data from the local file system to the distributed file system.

34. (Original) The service of claim 33, wherein the virtual storage server is a virtual object storage server.

35. (Canceled)

36. (Original) The service of claim 32, wherein the federation layer contains a distributed file system client for transferring and translating communications from a local application on the local computer system to the virtual server.

37. (Currently amended) The service of claim 32, further comprises an installation of the ~~virtual~~ file server in the local computer system.

38. (Currently amended) The service of claim 33, further comprises an installation of the virtual metadata server and the ~~virtual~~ file in the local computer system.

39. (Original) The service of claim 34, further comprises an installation of the virtual object storage server in the local computer system.

40. (Original) The service of claim 36, further comprises an installation of the distributed file system client in the local computer system.

41. (New) A processor-implemented service for federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

an addition of a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

an allowance of local applications to access both the data exposed in the local file system and data in other parts of the distributed file system;

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

an assignment of an object ID number to each object found in the local file system;

a detection of multiple hard links to a same file by comparing an inode number of files;

a use of a same object ID number for the multiple names for that object upon determination that one file has more than one hard link; and

a deletion of only the object when a last hard link has been unlinked.

42. (New) A computer program product having instruction codes stored on a computer-usable medium for federating a local file system into a distributed file system while preserving local access to an existing data in the local file system, comprising:

a set of instruction codes for adding a federation layer that allows both a local client and a plurality of distributed clients to access the existing data;

a set of instruction codes for allowing local applications to access both the data exposed in the local file system and data in other parts of the distributed file system; and

wherein the federation layer establishes a detour between the local applications and the local file system, to provide access to a distributed mechanism;

wherein the federation layer contains an object ID database for mapping between a file name and a unique object ID number;

a set of instruction codes for assigning an object ID number to each object found in the local file system;

a set of instruction codes for detecting multiple hard links to a same file by comparing an inode number of files;

a set of instruction codes for using a same object ID number for the multiple names for that object, upon determination that one file has more than one hard link; and

a set of instruction codes for deleting only the object when a last hard link has been unlinked.